Tecnológico de Costa Rica

Introducción a la programación

Profesores Diego Mora – Jeff Schmidt

Examen III – 2025-06-11

Tiempo 1 hora con 45 minutos

Instrucciones:

- Se le presentan 4 problemas para los cuales debe desarrollar en Python los programas que los solucionen. Los 4 problemas deben solucionarse **recursivamente**.
- Todas las funciones deben ser recursivas, por ejemplo, si requiere cantidad de dígitos, apariciones, está, debe programarlas de forma recursiva. La únicas que no serán recursivas serán las funciones base que llaman a su auxiliar recursiva.
- **Si el problema es sobre números, no puede utilizar listas** y **no puede convertir** los números **a string**.
- **No puede utilizar variables globales para ninguna solución.**
- Cada función debe retornar lo que se le solicita, **ninguno debe ser con prints o inputs**.
- Cada ejercicio tiene un valor de 25 puntos del examen.
- Durante los **primeros 30 minutos podrá hacer consultas** sobre el examen a los profesores o asistentes. Pasado este tiempo no se podrá consultar más.
- Utilice los nombres de la funciones **exactamente** como se le solicitan, en caso contrario no se le revisará el ejercicio.
- El uso de Internet está limitado exclusivamente para descargar y subir el examen del TEC Digital. El uso de internet, chats, modelos AI, Copilotos implicarán la cancelación de la prueba y obtendrá una calificación de cero, además del proceso disciplinario que el TEC indica para los casos de plagio.
- Para esta prueba NO podrá consultar algoritmos, programas o notas hechas en clase o de otras fuentes. Solo podrá acceder a hojas en blanco, lápiz y la computadora del laboratorio.

Entrega:

- Subir al TecDigital en Examenes > Examen III. El tiempo cierra a las 9:20 a.m. Suba un archivo .py con el nombre **EXIII_Apellido_Nombre.py**

1. detectar_valles (lista)

Desarrolle una función llamada detectar_valles(lista) que reciba como parámetro una lista de números enteros. La función debe identificar las secuencias de tres elementos consecutivos [anterior, actual, siguiente] donde se cumpla:

- **`El elemento actual es menor que el anterior y menor que el siguiente.`**

Estas secuencias se consideran **valles locales**.

La función debe retornar una **lista de listas**, donde cada sublista contiene los tres elementos consecutivos que forman un valle.

Condiciones:

- No se consideran el primer ni el último elemento por sí solos, pero pueden formar parte de una secuencia.
- Si no hay valles, retornar una lista vacía.
- Se pueden formar múltiples valles solapados, es decir, un elemento estar en dos valles.

Ejemplo 1:

entrada = [8, **4**, 5, **3**, 7, **1**, 2]

salida = [[8, 4, 5], [5, 3, 7], [7, 1, 2]]

Ejemplo 2:

entrada = [-3,10, **8**, 12, **9**, 15, 7]

salida = [[10, 8, 12], [12, 9, 15]]

Ejemplo 3:

entrada = [1,2,3,4,5,6]

salida = [ ]

2. es_montana (numero)

Programe una función llamada es_montana(numero) que reciba como parámetro un número entero positivo y determine si sus **dígitos forman una montaña numérica**.

Una **montaña** se define como una secuencia de dígitos que:

- **Sube estrictamente** (cada dígito mayor al anterior),
- Tiene un **pico** (el punto más alto, único),
- Luego **baja estrictamente** (cada dígito menor al anterior).

No se permiten mesetas (dígitos iguales seguidos) ni solo subida o solo bajada.

Resultado:

Si el número **sí forma una montaña**, la función debe retornar una **tupla con tres listas**:

- La lista de dígitos que forman la **subida** (sin incluir el pico),
- Una lista con el **pico**,
- La lista de dígitos de la **bajada**.

Si **no forma una montaña válida**, retornar: ([ ], [ ], [ ])

Ejemplos:

es_montana(19543) >>> ([1], [9], [5, 4, 3])

es_montana(12345) >>> ([], [], []) # Solo sube, no hay bajada

es_montana(975321) >>> ([], [], []) # Solo baja, no hay subida

es_montana(1**33**57931) >>> ([], [], []) #Tiene mesetas

es_montana(13531) >>> ([1, 3], [5], [3, 1])

3. cut(lista)

Escriba una función recursiva llamada cut(lista) que reciba una lista y produce una lista de listas, cortando cada sublista cuando aparecen números ceros.

```
>>> cut([1,23,0,29,2,0,1,0,34])
[[1, 23], [29, 2], [1], [34]]

>>> cut([1,2,0,0,0])
[[1, 2]]

>>> cut([1,2,3,4])
[[1, 2, 3, 4]]
```

---

4. elimine(num, dig)

Escriba un programa en Python llamado **elimine(num, dig)** que recibe un número entero positivo o cero y retorna un número sin los dígitos iguales al dígito dado, excepto el dígito más significativo, es decir, recorriendo el número de izquierda a derecha y eliminando los iguales en el número dado a excepción del que está más a la izquierda. No puede invertir el número ni convertirlo a lista, string u otro iterable.

Ejemplos:

> elimine(135**2**32, 2) retorna 13523

> elimine(**5**55, 5) retorna 5

> elimine(1**0**02010402, 0) retorna 102142